Minecraft on Z/OS

Aaron Kippins
Charlie Ropes
Brad Huntington
Michael Gildein

IBM

Contents

Today's presentation was based on an independent study run in conjunction with Marist College and their Masters Program.

Smooth Releases for Open World Games are Difficult to Attain

How Would Z/OS Fare Running This?

Our Goals

Get Minecraft on Z/OS

This is two-fold. We not only want to have a server running from Z/OS. We also would like a running ZCX Container.

This way we can get a feel for both running in a native environment and in a container.

Build Bots to Drive Stress

We want to have an easily reproducible case that we can run to drive work on all the different systems.

To do so we want to produce bots to explore the world and force terrain generation.

Collect Benchmarking Data

It's not good enough to just be running on Z. We also need to see where it positions compared to other platforms.

We're not only going to run on Z, but on Linux and a laptop as well to get a feel for the toll it takes on each platform.

Determine Z's Viability

With all of this we hope to be able to come to a synopsis of whether Z could potentially be a contender in this space.

Running the Experiment

Server on MacOS

Smoke testing bringing up a Minecraft server both in a docker container and native

Gathered Performance data using JProfiler.

Server on Ubuntu

Running a Minecraft server from a docker container.

Ran 20 exploration bots to produce stress.

Gathered Performance data using JProfiler.

Server on z/OS USS

Running a Minecraft server natively on a z/OS LPAR.

Ran 20 exploration bots to produce stress.

Gathered HIS data from the LPAR to find performance data.

Server on ZCX

Running a Minecraft server from a docker container using ZCX

Ran 20 exploration bots to produce stress.

Gathered HIS data from the LPAR to find performance data.

Test Creation

We designed bots to create stress on Minecraft servers using PrismarineJS/mineflayer as a code base to manage bot connections to servers.

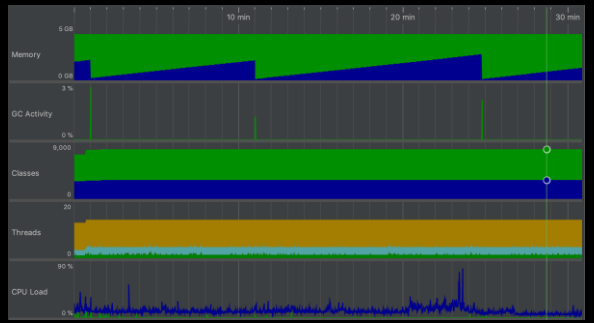
- Bots Login to a server and randomly select a location in the world to fly to.
- Bots rapidly generate new chunks on the map while exploring resulting in server stress.
- Bots select a new random location when they reach their destination. They repeat this action until shutdown so the server is constantly stressed.

Collecting Performance Documentation

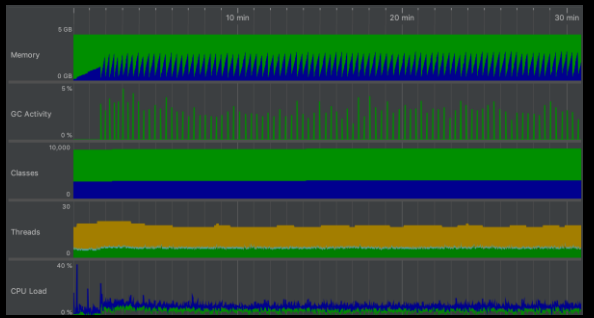
We utilized HIS data collection (z/OS) and JProfiler (Linux) to collect data on servers while idle and while stressed.

- HIS provides support for collecting CPU Measurement Facility data, which details counts of hardware events across the executing LPAR.
- JProfiler shows memory consumption, garbage collection activity, classes, active threads, and CPU load in real-time.

Idle



Stressed server



What We've Learned

Server Bring Up

Bringing up the servers was actually pretty trivial. Which in turn is a good thing and means that Z doesn't really hinder the possibility of running in that environment.

Creating the containers also was fairly normal. They just needed to be based on a Java image with the proper architecture type.

Dealing with Server Versioning

A lot of the issues that we ran into was dealing with the fact that the Vanilla instance isn't really optimized well no matter the environment.

Bot Building

Looking back the bots we have built are great for generating stress but they don't really mimic all the actions a player preforms when playing Minecraft.

Moving forward, being able to build bots that mimic actual players could help us better understand what capacity a server normally runs at.

Collecting Performance Data

It's hard to compare HIS data to data collected by JProfiler.

HIS data shows the use of the whole system while JProfiler only collects data on the profiled application.

HIS data displays metrics in an average over the course the run, while JProfiler shows a real-time graph of system metrics.

Our Minecraft Experience (Running with ZCX)

Z/OS Benefits and Value Proposition

Running With Z

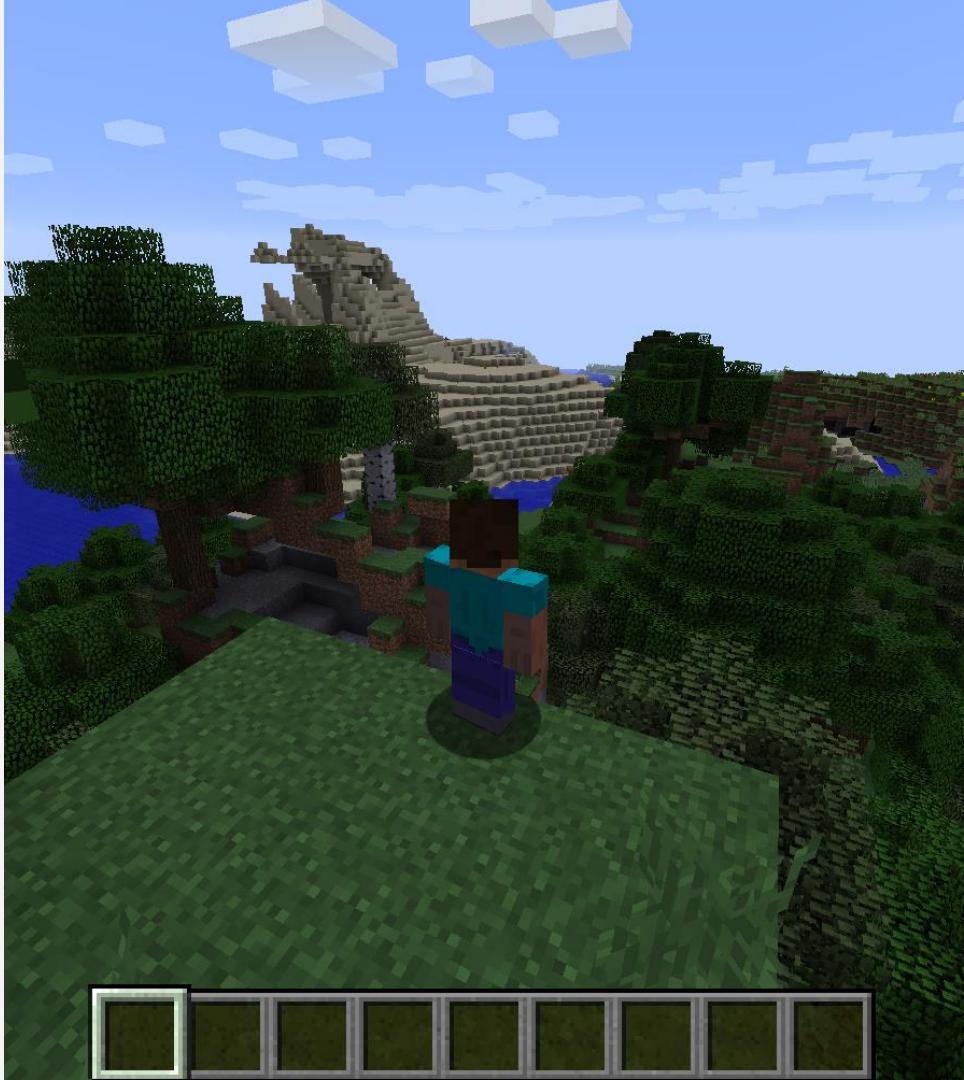
Running comparatively to being on a Linux Server there weren't many differences in the process. Playability wise they were pretty on par with each other.

During our study we didn't explicitly run with any optimization from the Z side. (Exploiting the middleware stack). Meaning that there's potential room for significant improvement.

Perceived Value

The gaming industry is massive and is growing rapidly.

We believe that there is an entire market that Z could be a contender in.



Next Steps

We'd like to continue looking into this. We'd like to attempt to implore using more of Z's Java specific resources and take a closer look a the the potential that running game servers could have.

Thank you

Aaron Kippins
Charlie Ropes
Brad Huntington
Michael Gildein
—
amkippin@us.ibm.com
caropes@us.ibm.com
huntingt@us.ibm.com
megildei@us.ibm.com